CSc 174 Database Management Systems

11. Query Processing and Optimization

Ying Jin
Computer Science Department
California state University, Sacramento

Query Execution Architecture

$$\begin{tikzcd}
\textit{Query} \arrow[d] \\
\text{Parser} \arrow[d] \\
\text{Preprocessor} \arrow[d] \\
\text{Logical query plan generator} \arrow[d] \\
\text{Physical query plan generator}
\end{tikzcd}$$

Parser and Preprocessor

- Parser
  - Define grammar
  - Parse Query in a high-level language
  - Generate parse tree
- Preprocessor
  - Replace the view with a parse tree that describes the view
  - Semantic checking

Logical Query Plans

- **Query tree**: a tree data structure that corresponds to a relational algebra expression.
- Leaf nodes - input relations
- Internal nodes - relational algebra operations

Physical Query Plans

- logical plan -> n physical query plans
- Algorithms to implement an operator (e.g. union, join)
- Intermediate results:
  - Materialized
  - Pipelined
- Cost_based plan Selection

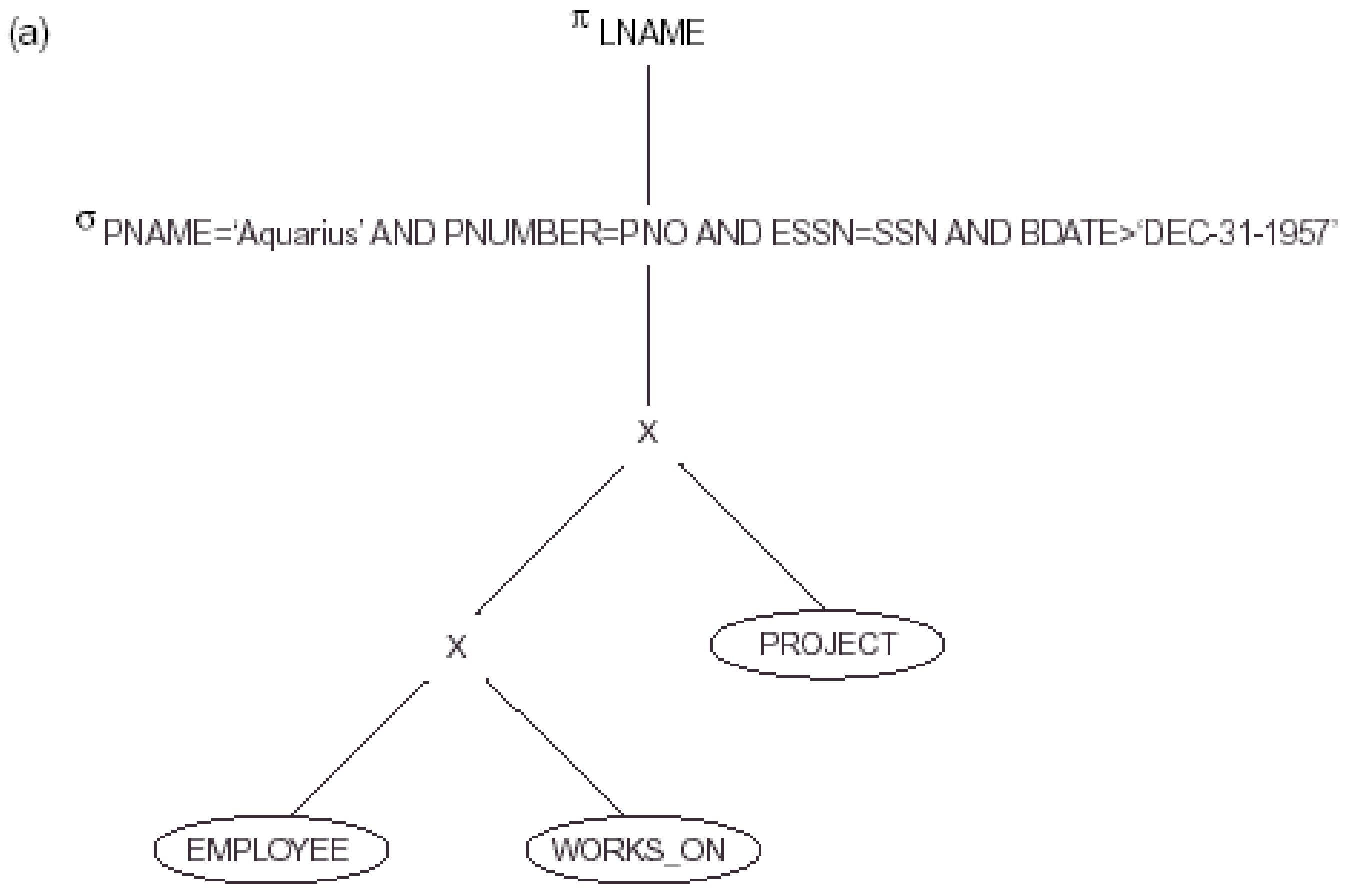
Using Heuristics in Query Optimization

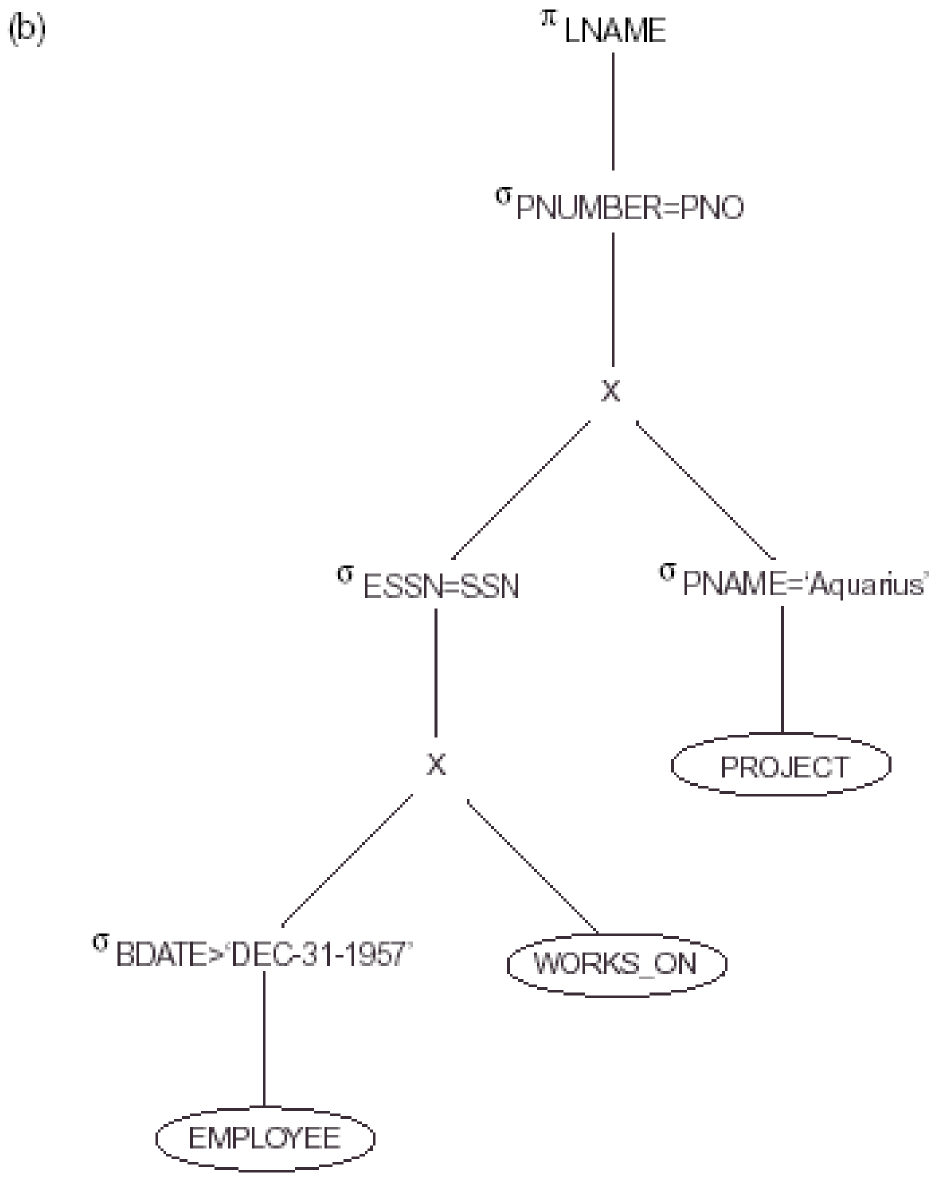
- Use logical query tree to present a query.
- Apply heuristic rules to convert a query tree into an equivalent query tree.
  - Represents a different relational algebra expression that is more efficient to execute
  - Gives the same result as the original one.

Query Tree Example

- Q:SELECT LNAME
  FROM EMPLOYEE, WORKS_ON, PROJECT
  WHERE PNAME = 'AQUARIUS' AND
  PNMUBER=PNO AND
  ESSN=SSN AND
  BDATE > '1957-12-31';
- Relational algebra presentation.

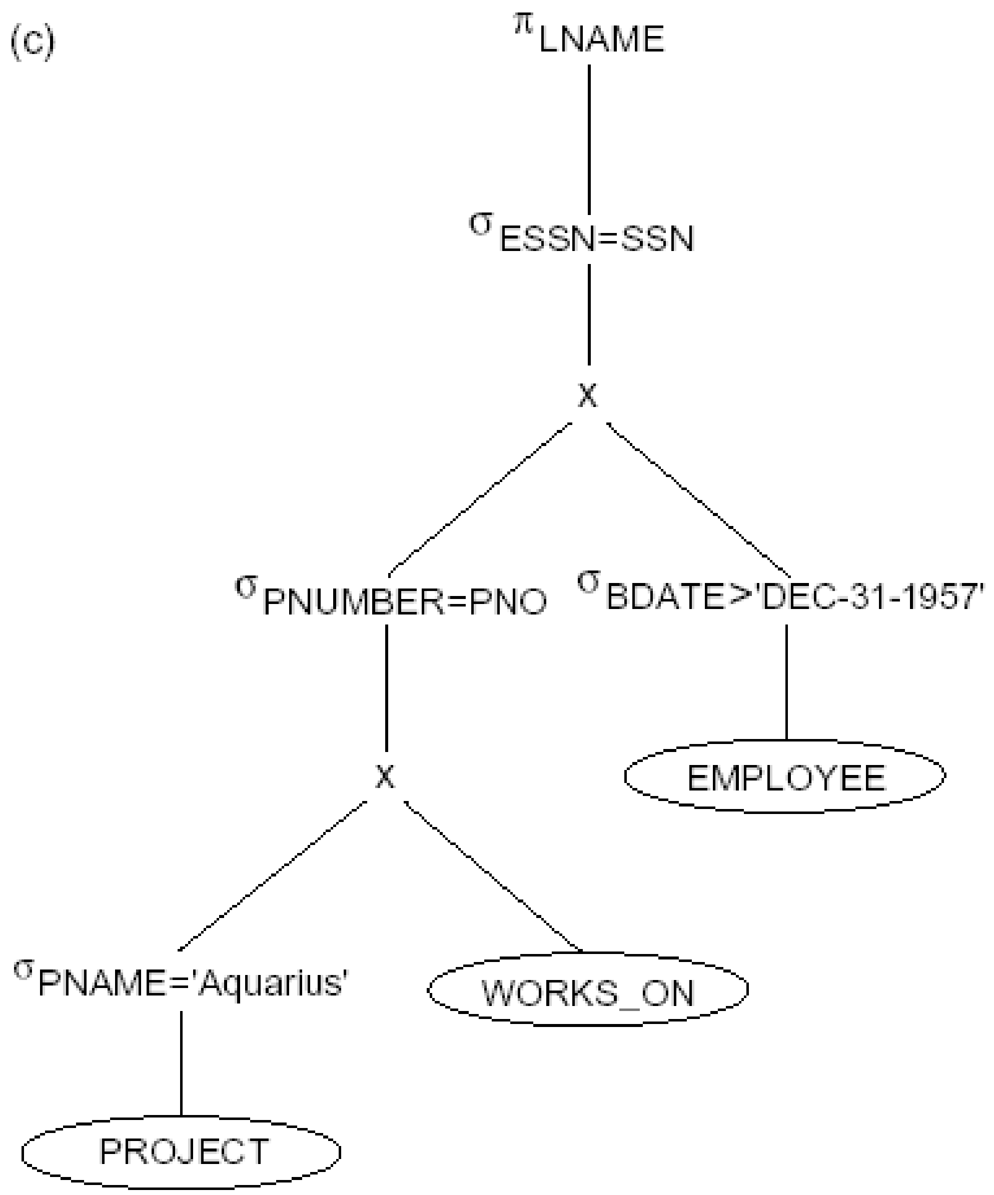
Query Tree for Q
- Initial (canonical) query tree


(a)
π LNAME
σ PNAME='Aquarius' AND PNUMBER=PNO AND ESSN=SSN AND BDATE>'DEC-31-1957'
X
X
PROJECT
EMPLOYEE
WORKS_ON

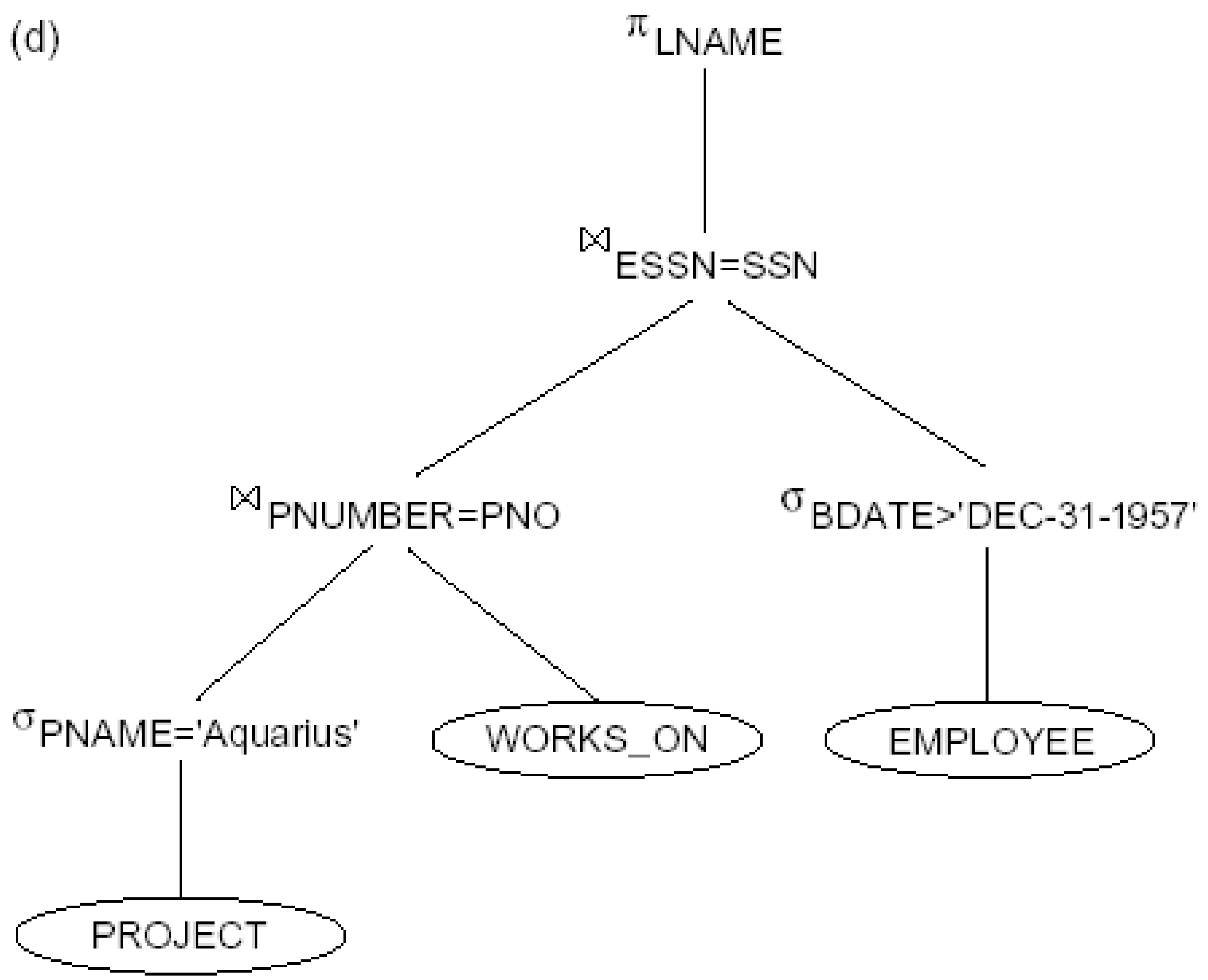


To reduce the number of tuples that appear in the CARTESIAN PRODUCT

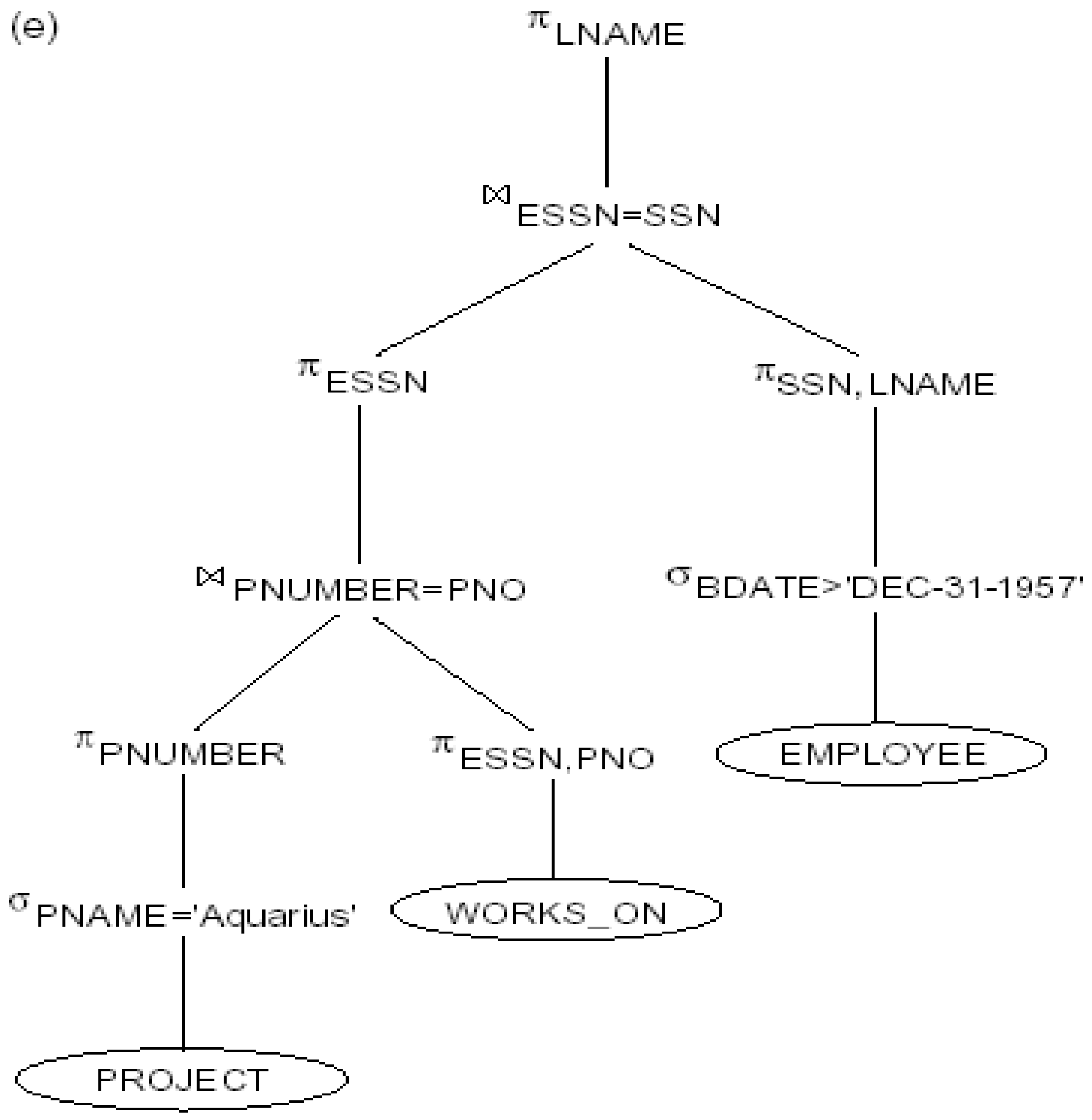
- Pname is unique.
- Produce less tuples in the first intermediate result.



- Replacing CARTESIAN PRODUCT and SELECT (essn-ssn) with JOIN operations



- Push down projection to reduce number of attributes of the intermediate results



Conclusion for Q?

- A query tree can be transformed step by step into another query tree that is more efficient to execute.
- Must make sure query optimizer must know which transformation rules preserve this equivalence.

General Transformation Rules for Relational Algebra Operations

1. Cascade of σ: A conjunctive selection condition can be broken up into a cascade (sequence) of individual s operations

   $\sigma_{\text{c1 AND c2 AND ... AND cn}}(R) =$

   $\sigma_{c1} (\sigma_{c2} (...(\sigma_{cn}(R))...))$

2. Commutativity of σ: The σ operation is commutative

   $\sigma_{c1} (\sigma_{c2}(R)) = \sigma_{c2} (\sigma_{c1}(R))$

General Transformation Rules for Relational Algebra Operations (Cont.)

3. Cascade of π: In a cascade (sequence) of π operations, all but the last one can be ignored:

$$\pi_{\text{List1}} (\pi_{\text{List2}} (...(\pi_{\text{Listn}}(R))...)) = \pi_{\text{List1}}(R)$$

4. Commuting σ with π: If the selection condition c involves only the attributes A1, ..., An in the projection list, the two operations can be commuted:

$$\sigma_c (\pi_{A1, A2, ..., An} (R)) = \pi_{A1, A2, ..., An} (\sigma_c (R))$$

General Transformation Rules for Relational Algebra Operations (Cont.)

5. Commutativity of $\bowtie$ (and x): The $\bowtie$ operation is commutative as is the x operation: $R \bowtie_c S = S \bowtie_c R$; $R \times S = S \times R$
6. Commuting σ with $\bowtie$ (or x): If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows :

   $\sigma_c (R \text{ X } S) = (\sigma_c (R)) \text{ X } S$

   Alternatively, if the selection condition c can be written as (c1 and c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:

   $\sigma_c (R \text{ X } S) = (\sigma_{c1} (R)) \text{ X } (\sigma_{c2} (S))$

General Transformation Rules for Relational Algebra Operations (Cont.)

7. Commuting π with $\bowtie$(or x): Suppose that the projection list is L = {A1, ..., An, B1, ..., Bm}, where A1, ..., An are attributes of R and B1, ..., Bm are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:

$$\pi_L (R \bowtie_C S) = (\pi_{A1, ..., An} (R)) \bowtie_C (\pi_{B1, ..., Bm} (S))$$

General Transformation Rules for Relational Algebra Operations (Cont.)

8. Commutativity of set operations: The set operations ∪ and ∩ are commutative but – is not.

9. Associativity of ⋈, x, ∪, and ∩ : These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have

$$(R \,\theta\, S) \,\theta\, T = R \,\theta\, (S \,\theta\, T)$$

10. Commuting σ with set operations: The σ operation commutes with ∪ , ∩ , and –. If θ stands for any one of these three operations, we have

$$\sigma_c (R \,\theta\, S) = (\sigma_c (R)) \,\theta\, (\sigma_c (S))$$

General Transformation Rules for Relational Algebra Operations (Cont.)

11. The π operation commutes with $\cup$.

    $\pi_L (R \cup S) = (\pi_L (R)) \cup (\pi_L (S))$

12. Converting a (σ, x) sequence into $\bowtie$ If the condition c of a σ that follows a x Corresponds to a join condition, convert the (σ, x) sequence into a $\bowtie$ as follows:

    $(\sigma_C (R \times S)) = (R \bowtie_C S)$

13. Other transformations

NOT (c1 AND C2) =(NOT c1) OR (NOT c2)

NOT (c1 OR C2) = (NOT c1) AND (NOT c2)

A Heuristic Algebraic Optimization Algorithm

1. Using rule 1, break up any select operations with conjunctive conditions into a cascade of select operations.
2. Using rules 2, 4, 6, and 10 concerning the commutativity of select with other operations, move each select operation as far down the query tree as is permitted by the attributes involved in the select condition.
3. Using rule 5 and 9 concerning associativity of binary operations, rearrange the leaf nodes of the tree so that the leaf node relations with the most restrictive select operations are executed first in the query tree representation. Make sure that the ordering of leaf nodes does not cause CARTESIAN PRODUCT operation.

Outline of a Heuristic Algebraic Optimization Algorithm (cont.)

4. Using Rule 12, combine a Cartesian product operation with a subsequent select operation in the tree into a join operation.
5. Using rules 3, 4, 7, and 11 concerning the cascading of project and the commuting of project with other operations, break down and move lists of projection attributes down the tree as far as possible by creating new project operations as needed.

Summary of Heuristics for Algebraic Optimization

1. The main heuristic is to apply first the operations that reduce the size of intermediate results.
2. Perform select operations as early as possible to reduce the number of tuples and perform project operations as early as possible to reduce the number of attributes. (This is done by moving select and project operations as far down the tree as possible.)
3. Replace Cartesian product with join when possible.

Examples

- More examples about query optimization using heuristics.

These slides are based on the textbook:

R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 7th Edition, Addison-Wesley.